Electron Commerce Res (2008) 8: 79–101
DOI 10.1007/s10660-008-9014-0

The concept of decentralized and secure electronic marketplace

Constantin Serban · Yingying Chen · Wenxuan Zhang · Naftaly Minsky

Published online: 12 April 2008

Abstract For commerce (electronic or traditional) to be effective, there must be a degree of trust between buyers and sellers. In traditional commerce, this kind of trust is based on such things as societal laws and customs, and on the intuition people tend to develop about each other during interpersonal interactions. The trustworthiness of these factors is based, to a large extent, on the geographical proximity between buyers and sellers. But this proximity is lost in e-commerce.

In conventional electronic marketplaces the trust among participants is supported by a central server which imposes certain trading rules on all transactions. But such centralized marketplaces have serious drawbacks, among them: lack of scalability, and high cost.

In this paper we propose the concept of *Decentralized Electronic Marketplace* (DEM) which allow buyers and sellers to engage in commercial transactions, subject to an explicitly stated set of trading rules, called the *law* of this marketplace—which they can trust to be observed by their trading partners. This trust is due to a decentralized, and thus scalable, mechanism that enforces the stated law of the DEM. We implement an electronic marketplace for airline tickets in order to illustrate the feasibility of the proposed concepts for decentralized and secure electronic marketplace.

C. Serban (✉) · X. Zhang · N. Minsky
Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854, USA
e-mail: serban@cs.rutgers.edu

X. Zhang
e-mail: wzhang@cs.rutgers.edu

N. Minsky
e-mail: minsky@cs.rutgers.edu

Y. Chen
Department of Electrical and Computer Engineering, Stevens Institute of Technology, Burchard 212, Hoboken, NJ 07030, USA
e-mail: yingying.chen@stevens.edu

Keywords Electronic commerce · Secure electronic marketplace · Distributed systems · Decentralization · Decentralized enforcement · Law governed interaction

For commerce (electronic or traditional) to be effective there must be a degree of trust between buyers and sellers. When buying an airline ticket, for example, the buyer needs an assurance that what he (or she) is getting is an authentic ticket, issued by the airline in question, and that it is not forged and cannot be duplicated. Also, if the payment for the ticket is done via a credit card, the buyer needs to trust the seller not to use the credit card for anything but the transaction at hand, and not to disclose it to anybody else. In traditional commerce, the trust between buyers and sellers is based on such things as societal laws and customs, and on the intuition people tend to develop about each other during interpersonal interactions. The trustworthiness of these factors is based, to a large extent, on the geographical proximity between buyers and sellers. It is the physical venue of the trade that is subject to specific trading laws, which may be enforced by the local police; and it is the eye contact between the trading partners that may induce a degree of trust between them.

But no such physical venue exists for the electronic marketplace. Indeed, the participants in electronic commerce might reside in different countries, and may be subject to different customs or habits. The trading partners are also invisible to each other, and are often immune from traditional kind of law enforcement. We need therefore some other, non-traditional, means for inducing trust in such a marketplace.

The conventional approach for electronic marketplaces is to use a centralized server, i.e. a single logical entity that mediates every transaction between buyers and sellers. Examples of such marketplaces include eBay (http://www.ebay.com), the Ford marketplace for automotive parts (http://www.pricingcentral.com/ford/), Open Market [1], and AuctionBot [2]. But although these particular marketplaces operate very effectively, the general concept of centralized electronic marketplace has several serious drawbacks. First, the participants in such conventional marketplace need to trust the trading rules employed by the server. But these rules are usually hard-coded into the code of the server, which is not generally available for inspection by the customers. And even if the code of the server is made available to the customers, it would not lend itself for serious analysis due to its size and complexity. Second, a centralized marketplace can achieve scalability—with respect to large number of participants and high transactions volume—only through massive replication, which tend to be very expensive.

In this paper we propose the concept of *Decentralized Electronic Marketplace* (DEM) which allows buyers and sellers to engage in commercial transactions, subject to an explicitly stated set of trading rules, called the *law* of this DEM. This law is enforced in a decentralized and inherently scalable manner, using a distributed set of trusted *controllers* that mediates all transaction subject to the law of the DEM. The mechanism provides buyers and sellers with sufficient degree of trust in each other such that they can engage in commercial trading. The trusted agents are generic in nature: they can accommodate multiple laws, concurrently serving multiple marketplaces, thus greatly reducing the cost of maintaining a server infrastructure.

A DEM is launched essentially by defining its law: this act has no real cost, because it does not involve the creation of any central mediator. Once launched, a DEM can grow incrementally in a scalable manner, simply by sellers and buyers joining in. In practice, such growth might require advertising, which is not discussed in this paper.

The concept of decentralized marketplaces is implemented via a decentralized control mechanism called Law Governed Interaction (LGI) [3, 4], which provides means for the specification of the law of a given DEM, and for its decentralized enforcement. The LGI mechanism has been fully implemented and has been recently released for public use [5, 6]. LGI has also been used for the implementation of the specific DEM that serves as an example in this paper. This implementation can be viewed as a proof of concept. But for the concept of DEM to be used in practice, by real buyers and seller all over the Internet, it is necessary for the middleware that supports LGI to be commercially deployed. Even though we do not see any technical impediments for such deployment, this issue is beyond the scope of our work. Note that a brief and incomplete version of this paper has been published in [7].

The rest of the paper is organized as follows. We briefly discuss some related work is in Sect. 1. Section 2 introduces a motivating example of a DEM designed for the trading of airline tickets. An outline of LGI, providing the computational basis for our concept of marketplace is presented in Sect. 3. Section 4 describes the implementation of the decentralized and secure marketplace for airline tickets introduced in Sect. 2, and we conclude in Sect. 5.

1 Related work

We have already pointed out that the conventional approach to electronic marketplaces is based on a central mediator. And we have explained the limitations of this approach, despite some very successful marketplaces of this kind, such as EBay. We will mention here the work most closely related to ours.

First, Schmees [8] in his 2003 paper "Distributed digital commerce," discussed the benefits of decentralized market for digital goods, and studied the processes involved in digital trading and their implementation using P2P communication. However, although Schmees admits the importance of trust and security in the marketplace, he did not propose any mechanism for achieving them. The DEM model proposed in this paper addresses exactly these issues.

Second, the European SEMPER project [3, 9] attempts to examine systematically the security requirements of electronic marketplaces, and proposed a framework for addressing them. The resulting open security architecture of SEMPER offers users the ability to select components of their choice from the SEMPER libraries, and to associate a certain level of trust with these chosen components. Before trading, SEMPER proposes a series of agreements to establish a set of rules for each role: buyer, seller, bank, certification authority, and etc. Users playing these roles can commit to abide by these rules. The agreement is signed on paper with a third party. It establishes in advance the liability of the parties regarding the future transactions, which they might want to conduct. The basic trust assumption of SEMPER has been that each user trusts his or her own machine, but not the machine of the partner. The client could only protect its user, not the user on the other end of the wire. However, the SEMPER project proposed no practical implementation, and had no continuation after the project has been completed in 2000.

Third, PeerTrust [10, 11] presents a trust model based on reputation in order to facilitate trust and minimize the risk involved in e-commerce transactions. The authors discuss different factors that have to be taken into account when computing a reputation, such as the feedback from other peers, the number of transactions and the credibility of the feedback providers, as well as context factors peculiar to the transaction in particular and the community in general. This approach, however, like any other reputation approach, is entirely reactive. That is, it minimizes the risk of engaging dishonest traders, without providing any protection towards a certain outcome of a given transaction. Our paper presents a protection mechanism based on proactive enforcement of individual transactions.

Fourth, in a project closely related to ours, which also employs LGI, a Decentralized Peer-to-Peer Auction framework [12] has been proposed. But it is only focused on the electronic auctions, and it has been designed essentially as a sellers' mechanism: it enables the sellers to achieve flexibility, by offering different auctioning policies, and trading transparency, by making these policies explicit and trustworthy for the buyers. Our present work, in contrast, provides a common set of rules that are enforced for all the participants (buyers and sellers alike), such that they can trust each other, effectively creating the necessary glue that holds a distributed electronic marketplace together.

Finally, we previously introduced the idea of DEM in [7], where we have proposed a preliminary and incomplete set of rules that are to govern a DEM. In [7] we provided only a brief outline of how these rules are to be implemented. In contrast, this paper offers a meaningful discussion and a complete implementation for a revised set of rules. This revised set of rules contains new mechanisms for establishing the long-term reputation tracking and to address the issue of conformance to the governmental laws. We consider that these rules are not only useful, but they are essential for any realistic implementation of a decentralized electronic marketplace.

Our work is novel in that we have proposed the concepts of decentralization and security for general electronic marketplaces. In addition, our work differs from most previous research in that we have a complete implementation of the proposed architecture and we have released publicly the infrastructure, *Moses*, which can be used for realizing this concept in practice.

2 A marketplace for airline tickets—a motivating example

In order to show how a decentralized electronic marketplace (DEM) might operate, let us consider a specific such marketplace for airline tickets, to be called the *Airline Ticket Marketplace (ALTM)*. We first present the various participants in this marketplace, we explain broadly their roles, and we present the trading rules that govern them.

The participants in this marketplace, which might be widely distributed and are assumed to communicate with each other via the Internet, are as follows: (1) The *airlines*, that issue tickets, in an electronic form, (2) The *banks*, that represent the financial infrastructure facilitating the payments for the tickets (for simplicity, we assume that all payments are to be done via credit cards, and the banks provide credit card authorization and money transfer services for the trading partners), (3) The *Sellers*, that obtain tickets from the airlines, (4) The *Buyers* who purchase tickets from the sellers, (5) A *reputation server* that helps to maintain the long-term reputation of the sellers (the usage and the role of the reputation server will become clear later in Sect. 4), and (6) a *certification authority CA* that provides digital credentials to the various participants in the DEM, except for the buyers.

In order to provide for a secure and trustworthy marketplace, we identify the following requirements that are to govern the interaction and the conduct of these participants. We will refer to these rules of conduct as *the policy* of the ALTM marketplace. This policy, to be presented here only in broad terms, is concerned with the following areas:

1. *Authentication of identity:* Sellers, airlines, banks, and the reputation server are required to authenticate themselves by presenting a digital certificate signed by the specified certification authority. Buyers are not required to authenticate themselves.
2. *Authenticity of the tickets:* Tickets sold by sellers are required to be authentic. That is, sellers can only sell tickets obtained from an airline, and they should not be able to duplicate such tickets.
3. *Security and privacy of credit card payment:* Payments under this marketplace are to be made via credit cards, with the following guarantees to the buyers: (a) the credit card would be charged only for the cost of the purchased airline ticket, and only once; and (b) no information about the credit card being used would be given to anybody but the specified bank, not even to the seller itself.
4. *Reputation services:* The reputation of the sellers has to be tracked and reported, in a decentralized manner, more about which later.
5. *Conformance to the various governmental laws:* The trading conduct of the buyers and sellers should be subject to various national or international laws that have jurisdiction over the marketplace.

The various points of this policy will be discussed in more detail, as part of the LGI law that is to govern this marketplace.

Figure 1 is a schematic depiction of the interactions taking place in the ALTM marketplace. The certification authority, the airlines, and the reputation server are loosely coupled with the rest of the participants, as depicted by the discontinued arrows. The certification authority provides digital certificates to all the participants (except for the buyers) initially, in an offline manner; the airlines transfer tickets to the sellers asynchronously, with no direct involvement in the transactions; and the reputation server saves and retrieves the reputation of the sellers, only as they become active or inactive.

The actual sale of tickets proceeds as follows. A buyer submits a request to a seller, containing the *id* of the ticket it wishes to purchase, and the credit card information that is to be used for the payment (step 1). When the seller receives this request, it will provide the credit card information to the bank (step 2), which in turn will confirm (or deny) the transaction (step 3). When the seller receives the confirmation,

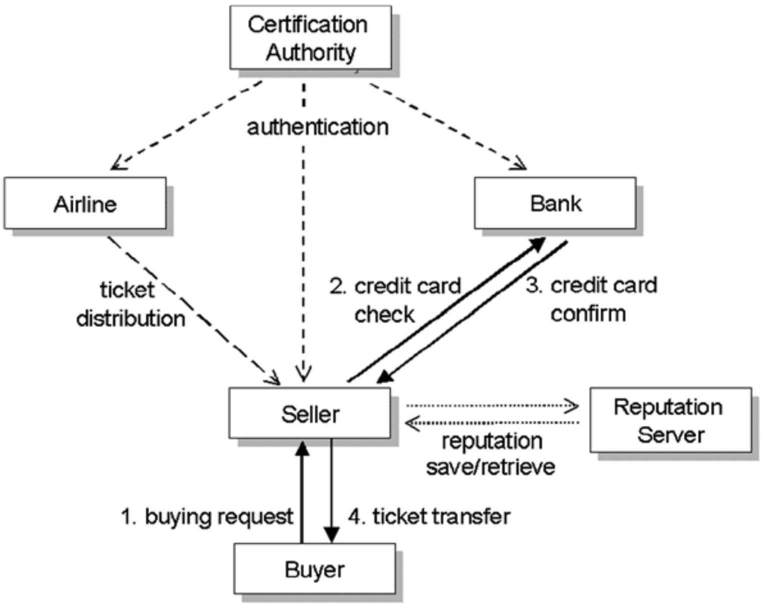

Fig. 1 Airline ticket marketplace (ALTM)

it will provide the buyer with the e-ticket that has been previously obtained from the airline (step 4). Note that the only entity involved in the actual transaction beside the buyer and the seller is the bank, which, as in the case of traditional marketplaces, provides credit card payments. Beside this aspect, the interaction in this marketplace is decentralized: the buyers and the sellers exchange tickets without involving any other centralized entity. Would we have chosen payments via digital cash [1], which is easy to implement using our LGI mechanism (as shown in [13]), the interaction would be entirely decentralized.

Note also that beside the money exchange mentioned above, the participants in the marketplace are likely to exchange a variety of other information messages that need not be regulated, such as advertisement of goods, search and retrieval of information, formal or informal negotiation processes, and so on. We are not concerned with these types of messages. In Sect. 4, we will show how the marketplace can be implemented in a largely decentralized manner by simply enforcing the ALTM policy, using the LGI mechanism.

3 An overview of LGI

To build a decentralized and secure marketplace, we use the LGI paradigm in order to define the regulation policies as *laws*. The most salient and unconventional aspects of LGI laws are their strictly local formulation and the decentralized nature of their enforcement. In this section, we provide an overview of the LGI concepts and architecture.

LGI is a mode of interaction that allows an open group of distributed heterogeneous agents[1] to interact with each other with confidence that the explicitly specified policies, called the *law* of the open group, is complied with by everyone in the group [4, 14]. The messages exchanged under a given law L are called L-messages, and the group of agents interacting via L-messages is called a *community* C, or more specifically, an L-community C_{L}.

The concept of "open group" has the following semantic: (a) the membership of this group can be very large, and can change dynamically; and (b) the members of a given community can be heterogeneous. LGI does not assume any knowledge about the structure and behavior of the members of a given L-community. All such members are treated as black boxes by LGI. LGI only deals with the interaction between these agents. Members of a community are not prohibited from non-LGI communication across the Internet, or from participation in other LGI-communities.

For each agent x in a given L-community, LGI maintains the *control-state* CS_x of this agent. These control-states, which can change dynamically, subject to law L, enable the law to make distinctions between agents, and to be sensitive to dynamic changes in their states. The semantic of the control-state for a given community is defined by its law, and could represent such things as the role of an agent in this community, its privileges and reputation. The CS_x is a bag of objects called *Terms*. For instance, under the *ALTM Law* to be introduced in Sect. 4, a *Term* with the value *role(airline)* in the control state of an agent denotes that the agent has been authenticated to be a genuine airline.

LGI is currently implemented by the *Moses* toolkit. The software, its supporting documentation, and an online infrastructure for public access are available for free on its website at http://www.moses.rutgers.edu.

In the rest of this section we will continue to further discuss the concept of law, its local nature, and we will present a description of the decentralized mechanism for law enforcement. The concepts of *obligations* and the treatment of *certificates* will be elaborated briefly. We do not discuss here several important aspects of LGI, including the *interoperability* between communities, the treatment of *exceptions*, the deployment of L-communities, and the performance of its current implementation. For a complete understanding of these issues, the reader is referred to [14, 15].

3.1 The concept of law and its enforcement

Generally speaking, the law of a community C is defined over certain types of events occurring at members of C, mandating the effect that any such event should have; this mandate is called the *ruling* of the law for a given event. The events subject to laws, called *regulated events* include (among others): the *sending* and the *arrival* of an L-message; the *coming due* of an *obligation* previously imposed on a given object; and the submission of a *digital certificate*. The operations that can be included in the ruling of the law for a given regulated event are called *primitive operations*. They include, operations on the control-state of the agent where the event occurred

[1] Given the popular usages of the term "agent", it is important to point out that we do not imply by it either "intelligence" or mobility, although neither of these is being ruled out by this model.

(called, the "home agent"); operations on messages, such as *forward* and *deliver*; and the imposition of an obligation on the home agent. Note that the ruling of the law is not limited to accepting or rejecting a message, but can mandate any number of operations, like the modifications of existing messages, and the initiation of new messages and of new events, thus providing the laws with a strong degree of flexibility. More concretely, LGI laws are formulated using an *event-condition-action* pattern. In this paper we will depict a law using the following pseudo-code notation:

```
upon <event>  if  <condition>
              do  <action>
```

Where the <*event*> represents one of the regulated events, the <*condition*> is a general expression formulated on the event and control state, and the <*action*> is one or more operations mandated by the law. This definition of the law is abstract in that it is independent of the language used for specifying laws. Concretely we use two such languages: one is based on Prolog, and the other one on Java. However, despite the pragmatic importance of a particular language being used for specifying laws, the semantics of LGI is basically independent of that language.

Thus, a law L can regulate the exchange of messages between members of an L-community, based on the control-state of the participants; and it can mandate various side effects of the message exchange, such as modification of the control states of the sender and/or receiver of a message, and emission of extra messages.

3.1.1 The local nature of laws

Although the law L of a community C is *global* in that it governs the interaction between *all* members of C, it is enforced locally at each member of C. This is accomplished by the following properties of LGI laws:

- L only regulates local events at individual agents.
- The ruling of L for an event e at agent x depends only on e and the local control-state CS_x of x.

The ruling of L at x can mandate only local operations to be carried out at x, such as an update of CS_x, the forwarding of a message from x to some other agent y, and the imposition of an obligation on x. The fact that *the same law* is enforced at all agents of a community gives LGI its necessary global scope, establishing a *common* set of ground rules for the members of C and providing them with the ability to trust each other, in spite of the heterogeneity of the community. Furthermore, the locality of law enforcement enables LGI to scale with the size of the community.

3.1.2 Distributed law-enforcement

Broadly speaking, the law L of community C_L is enforced by a set of trusted agents, called *controllers*, that mediate the exchange of L-messages between members of C_L. Every member x of C has a controller T_x assigned to it (T here stands for trusted agent) which maintains the control-state CS_x of its client x. All these

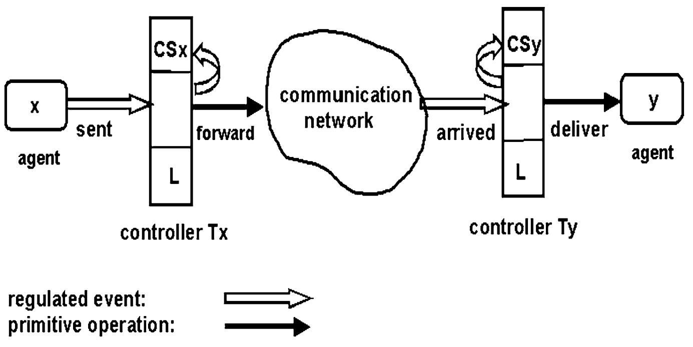

Fig. 2 Enforcement of LGI through controllers

controllers, which are *logically* placed between the members of C and the communication medium as illustrated in Fig. 2 carry the *same law* L. Every exchange between a pair of agents x and y is thus mediated by *their* controllers T_x and T_y, so that this enforcement is inherently decentralized. However, several agents can share a single controller, if such sharing is desired. The efficiency of this mechanism, and its scalability, are discussed in [14]. Controllers are generic, and can interpret and enforce any well- formed law. A controller operates as an independent process, and it may be placed on any machine, anywhere in the network. We have implemented a *controller-service*, which maintains a set of active controllers. To be effective in a widely distributed enterprise, this set of controllers need to be well dispersed geographically, so that it would be possible to find controllers that are reasonably close to their prospective clients.

3.1.3 The basis of trust between members of a community

For members of an L-community to trust its interlocutors to obsere the same law, one needs the following assurances: (a) Messages are securely transmitted over the network; (b) The exchange of L-messages is mediated by controllers interpreting the *same law* L; and (c) All these controllers are correctly implemented. If these conditions are satisfied, then it follows that if agent y receives an L-message from agent x, this message must have been sent as an L-message; in other words, that L-messages cannot be forged.

Secure transmission is carried out via traditional cryptographic techniques. To ensure that a message forwarded by a controller T_x under law L would be handled by another controller T_y operating under the same law, T_x appends the one-way hash [16] H of law L to the message it forwards to T_y. T_y would accept this as a valid L-message if and only if H is identical to the hash of its own law.

As to the correctness of controllers, we assume here that every L-community is willing to trust the controllers certified by a given certification authority (*CA*), which is specified by the law L. In addition, every pair of interacting controllers must first authenticate each other by means of certificates signed by this *CA*. This requires the existence of a trusted set of controllers, maintained by what we call a *controller-service*, or *CoS*, to be discussed below.

3.1.4 The controller-service (CoS) of LGI

The controller service is responsible for maintaining a reliable and secure set of controllers, which collectively constitute the trusted computing base (TCB) of LGI. We have designed and constructed a prototype of such CoS, which maintains a set of continuously tested, and geographically distributed controllers, and which provides the services of these controllers to agents who want to operate under LGI, as follows. For an agent x to be able to exchange L-messages with other members of an L-community, it must: (a) procure an LGI controller from a trusted CoS; and (b) notify this controller that it wants to use it under law L.

For use within an enterprise, such a CoS can be maintained and managed by the enterprise administration, and can thus be trusted by all enterprise computations. But for the CoS to support a marketplace, to be used by people and servers distributed all over the Internet, and not belonging to any single administrative domain, the CoS needs to function as a *public utility*. There are no serious technical impediments to the construction of a CoS public service. But it needs to be done by a large financial or governmental organization that can serve as a trusted third party, with no financial interest in the computing activities regulated by its controllers. This organization must assume certain liabilities for various failures of the controllers provided to its customers. And it needs to provide audit trail of its controllers' activities, which are secure enough to be accepted in a court of law, in case of a dispute. The construction of such a public utility of controllers is beyond the scope of our present work.

3.2 Some advanced features of LGI

We introduce here briefly some of the advanced features of LGI, in particular those employed in this paper. For additional information about these features, and for a study of their use, the reader is referred to the LGI manual [5].

3.2.1 The treatment of certificates

The conventional usage of certificates includes: authentication of the identity of an agent; authentication of the role a given agent plays in a certain community; and testimonial of certain rights that a given agent obtained from another via *delegation*.

Certificates may be required by a given law L to certify the controllers used to interpret this law. Certificates may also be submitted by an actor A_x to its controller T_x. The effect of such certificates is subject to the law in question. Typically, such submitted certificates are used to authenticate the identity of the actor, or the role it plays in the environment in which the community in question operates.

For now, *Moses* supports SPKI/SDSI model [17] for certificates. But it would be very easy to adapt LGI to any other structure one may prefer. Under LGI, a certificate is a four-tuple (*issuer*, *subject*, *attributes*, *signature*), where *issuer* is the public-key of the *CA* that issued and signed this certificate, *subject* is the public-key of the principal that is the subject of this certificate, *attributes* is what is being certified about the *subject*, and the *signature* is the digital signature of this certificate by the *issuer*. The *attributes* field is essentially a list of (attribute, value) pairs. For example, the attributes of a certificate might be the list [name(johnDoe), role(seller)], asserting that the name of the subject in question is John Doe and its role in this community is a seller.

3.2.2 Enforced obligation

Informally speaking, an obligation under LGI is a kind of *motive force*. Once an obligation is imposed on an agent—generally, as part of the ruling of the law for some event at it—it ensures that a certain action (called *sanction*) is carried out at this agent, at a specified time in the future, when the obligation is said to come due, and provided that certain conditions on the control-state of the agent are satisfied at that time. Note that a pending obligation incurred by agent x can be *repealed* before its due time. The circumstances under which an agent may incur an obligation, the treatment of pending obligations, and the nature of the sanctions, are all governed by the law of the community.

3.2.3 The treatment of exceptions

Primitive operations that initiate messages, like deliver and forward, may end up not being able to fulfill their intended function. For example, the destination agent of a *forward* operation may fail by the time the forwarded message arrives at it. Such failures can be detected and handled via a regulated event called *an exception*, which is triggered when a primitive operation that initiates communication cannot be completed successfully. It is up to the law to prescribe what should be done to recover from such an exception. The syntax of an exception event is: *exception*(*op*, *diagnostic*) where *op* is the primitive operation that could not be completed, and *diagnostic* is a string describing the nature of the failure. The home of the exception event is the home of the event that attempted to carry out the failed operation. For instance, if a message m, forwarded by an agent x to an agent y operating under law L cannot reach its destination, then an event *exception*(*forward*($x, m, [y, L]$), "*destination not responding*") would be triggered at x. Commonly, exceptions are triggered by the *forward* and *deliver* primitive operation, as well as other communication primitives. More details about the exception mechanism are given in the LGI manual [5].

3.2.4 The hierarchical organization of laws

LGI provides a mechanism to organize the laws into hierarchies [18]. Each such hierarchy, or tree, of laws $t(L_0)$, is rooted in some law L_0. Each law in $t(L_0)$ is said to be (transitively) subordinate to its parent, and (transitively) superior to its descendents. Generally speaking, each law L' in a hierarchy $t(L_0)$ is created by refining a law L, the parent of L', via a delta L', where a delta is a collection of rules defined as a refinement of an existing law. The root L_0 of a hierarchy is a normal LGI law, except that it is created to be open for refinements, using a *consulting* function. This function allows the root law to suggest (pseudo) events to its subordinate delta, and to receive, and possibly interpret a proposed ruling. The final decision about the ruling of law L' is made by its superior law L, leaving its deltas only an advisory role. Thus, the process of refinement is defined in a manner that guarantees that every law in a hierarchy conforms (transitively) to its superior law.

4 Implementation of the airline ticket marketplace

Recall that a decentralized marketplace (DEM), as presented earlier in the introduction, is defined by the law that regulates the interaction between its participants, and not by any physical locus where these participants might meet.

In this section we define a particular such DEM, the airline ticket marketplace (ALTM), and we show the formal implementation of its law, called *ALTM law*, based on the rules of the ALTM policy introduced informally in Sect. 2.

Note that there is no set procedure or mechanism for writing such a law. The *ALTM law*, in particular can be written by a coalition of airlines that decide to establish common rules for the market for their tickets; it could be the result of a negotiation process among prospective vendors; or it can be developed by independent specialists in the field. Once such a law is written and published, it is a matter of a social process for it to be widely accepted by airlines, by vendors and by clients. We have nothing original to say about such a social process, except to point out that a prospective participant in this marketplace can examine its law before deciding to adopt it, or have the law examined by a consultant who understands LGI. Thus, the participants in an LGI-based marketplace can be much better informed about its rules than is the case with most traditional or web-based marketplaces.

Also note that, for simplicity, the *ALTM law* presented here is written under a *no failure assumption*. Failures can, and should be treated via the exception handling facility of LGI, described briefly in Sect. 3.2.3. An example of how these failures could be treated in LGI is provided in [19] and the LGI manual [5].

The rest of this section presents the LGI implementation of the ALTM law. This law is presented here incrementally, as a sequence of fragments, all specified in pseudo-code. The actual implementation of the law can be found at: http://www.moses.rutgers.edu/examples/marketplace/trade.law, for the Prolog implementation, and at http://www.moses.rutgers.edu/examples/marketplace/trade.java1, for the Java implementation.

4.1 Authentication of identity

In the ALTM marketplace, airlines, banks, and the reputation server have to be authenticated since they have a unique and critical role in the proper functioning of the market. Sellers also have to be authenticated because the airlines would not provide tickets to anonymous entities. In ALTM marketplace, the certification authority *CA* provides sellers, airlines, banks and the reputation server with *digital certificates* that carry their role and their identity.

The fragment of *ALTM Law* in Fig. 3 shows how the authentication of identity takes place. When a participant engages in the marketplace, it does so by sending a special *adoption* message to its LGI controller, message that can carry its certificate.

```
/* When an actor adopts a controller without supplying a digital certificate, its control state will contain role(buyer). */

R1: upon adopted(Args)
                              do Add(role(buyer))


/* Other participants in the marketplace are required to provide a digital certificate. After the certificate is verified, the
corresponding role (airline, bank, or reputation server, seller) is added into the CS. */

R2: upon adopted(Self, Issuer, Subject, Attributes, Args)
                              if (Subject != Self || Issuer ! = CA)
                                   return
                              if (Attributes.role = airline || bank || reputationServer)
                                   do Add(role(Attributes.role))
                                   do Add(name(Attributes.name))
                              if (Attributes.role = seller)
                                   do Add(name(Attributes.name))
                                   do forward(Self, get_reputation(Attributes.name),RS)
```

Fig. 3 The authentication of identity: fragment of ALTM law

When this message arrives at the controller, it invokes an *adopted* event. If no certificate is provided in this message, the agent will be assigned automatically a role of buyer, as shown in Fig. 3, Rule 1. If the actor presents a certificate, the controller verifies it against the public key of the CA, and it challenges the private key of the subject; subsequently Rule 2 in Fig. 3 is invoked. This rule proceeds as follows. If the *Subject* is not the presenter, or if the *Issuer* is not the *CA*, then no role and no identity are assigned to this agent: the ruling of the law is empty. If the *attributes* of the certificate contain the role of airline, bank or reputation server, then this role and the identity (i.e. the name) of the agent are extracted from the attributes and saved in the control state maintained by the controller on behalf of this agent. If the role in the certificate is seller, then its identity is saved in the control state, but not the *role*(*seller*). Instead, a *get_reputation*(*Attributes.name*) message is sent to the reputation server, for reasons to be discussed later in Sect. 4.5.

4.2 Ticket authenticity

Our policy requires that every ticket sold by a seller be authentic, i.e. issued by the specified airline. In ALTM marketplace this property is achieved using the interaction process depicted in Fig. 4. Conceptually, whenever an airline distributes its tickets, the seller's controller T_s maintains a copy of each ticket in its control state (more precisely, its hash).[2] Whenever such a ticket is sold, its authenticity is established by comparing the ticket against the copy previously stored in the control state.

The fragment of the *ALTM Law* that handles this process is presented in Fig. 5. Rule 1 ensures that only properly authenticated airlines can distribute tickets (recall from Sect. 4.1 that once an airline has authenticated itself to its controller, a term *role*(*airline*) is stored in its control state). When such a ticket arrives at a seller, the

[2] In practice, an airline ticket contains significant amounts of data such as serial number, date and time, source, destination, class, etc. This data can overload the control state CS_s if a large number of tickets are sold at the same time. In order to address this, we employ a one-way hash function to maintain a digest of the ticket in the control state instead of the full ticket

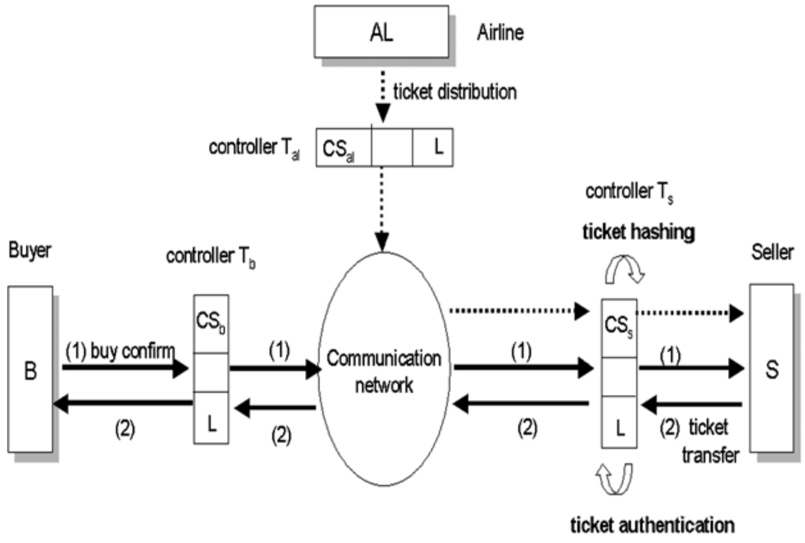

Fig. 4 Airline ticket distribution

```
/* Only an agent that has the role(airline) can distribute tickets.*/

R1: upon sent(Al, distribute(Ticket), S)
        if (CS has role(airline))
            do Forward


/* when a distribute(ticket) message arrives at a seller, hash(ticket) is added into its CS.*/

R2: upon arrived(Al, distribute(ticket), S)
        if (CS has role(seller))
            do Add(hash(ticket))
            do Deliver


/* When the seller sells an airline ticket (ticket_transfer(ticket)), the hash of the ticket is compared against the hash previously
stored in the CS of T_s. Upon success, T_s sends the ticket and removes the ticket hash from the CS, otherwise a warning message is
sent back to the seller.*/

R3: upon sent(S, ticket_transfer(ticket), B)
        if not(CS has role(seller))
            return
        if not(CS has hash(ticket))
            do Deliver("ticket_not_authentic")
        else
            do Remove(hash(ticket))
            do Forward
```

Fig. 5 Airline ticket authenticity: fragment of ALTM law of

one-way hash of the ticket is computed and stored in the control-state of T_s (Rule 2). At a later time, when the seller sells the ticket by sending a *ticket_transfer* message, the hash of the ticket being sold is computed (Rule 3). This value is compared against the value previously stored in the control state. If they match, the ticket can be sent to the buyer, and the hash value is removed from the sellers control state.

This scheme prevents a number of possible frauds. If the seller modifies a certain piece of information in the ticket, such as its class, T_s will detect the hash mismatch and will prevent such a message to be sent by the seller.

Also, a dishonest seller cannot sell a ticket more than once, because the hash value of the ticket is removed from T_s after a ticket is sold the first time. Note that the authenticity of tickets is ensured in a completely decentralized manner: the controller of each seller maintains the trusted copy of the ticket (i.e. its hash), and the airline is only involved in the initial ticket distribution phase, and not in the direct trading path.

Note that in the absence of the LGI mechanism, such forgery can be prevented in the following manner: for every bought ticket, the buyer can verify with the airline that the ticket is authentic and it has not been previously sold. This solution, however, introduces the airline in the direct trading path, impeding the decentralized nature of the marketplace. Another centralized, thus unscalable solution is to employ a trusted third party holding the ticket in order to verify and enforce its validity.

4.3 Security and privacy of credit card payment

As discussed in Sect. 2, the ALTM policy makes the following guarantee to a buyer: (a) its credit card would be charged only for the cost of the purchased airline ticket, and only once; and (b) no information about the credit card being used would leaked to the seller itself. In ALTM marketplace, the controller T_s of the seller protects the buyer's confidentiality by maintaining the credit card information without disclosing it to the seller, as presented in Fig. 6.

T_s will perform the credit card authorization on behalf of the seller by contacting the corresponding bank. Figure 7 presents the fragment ALTM law that controls this process. In Rule 1, whenever the buyer sends a *buy_confirm* message to the seller, T_s will send automatically a *credit_ck_req* (credit card checking request) to the bank

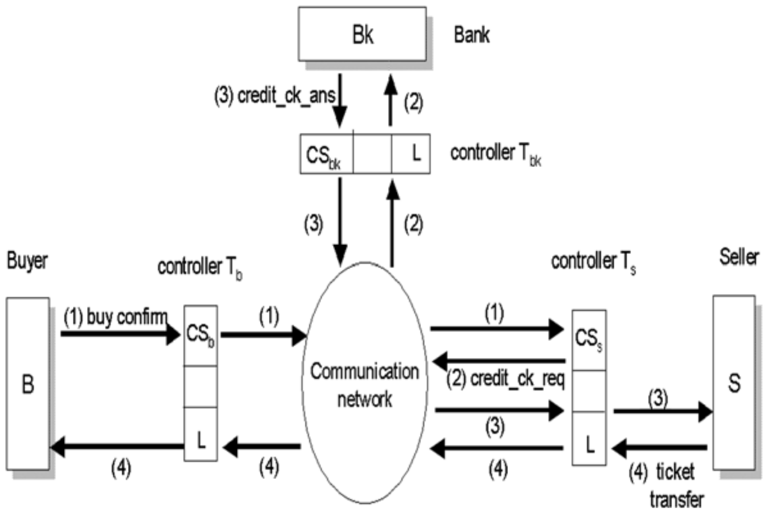

Fig. 6 Security and privacy of credit card payment

```
/* When a buy_confirm message arrives at the controller of the seller, an automatic credit check request is forwarded to the
bank, and an order information is saved in the CS*/

R1: upon arrived(B, buy_confirm(ticket, card), S)
        if (CS has role(seller))
            do Save(order(ticket))
            do Forward(B, credit_ck_req(card,tickt.price), Bank)


/* If an approved credit_ck_ans message arrives from the bank, the message is delivered to the seller to complete the
transaction. If the credit card transaction is not approved, a failure message is reported to the client automatically.*/

R2: upon arrived(Bank, credit_ck_ans, S)
        if not (CS has role(seller))
            return
        if (credit_ck_ans.value == approved)
            do Update(order(ticket))
            do Deliver(approved(order(ticket))
        else
            do Remove(order(ticket))
            do Forward(S,  not_approved,B)
```

Fig. 7 Security and privacy of credit card: fragment of ALTM law

and at the same time maintain a custom order record (*order*(*ticket*)) in its control state. After the bank performs the transaction, a *credit_ck_ans* message is sent to the seller. When this message arrives at T_s, Rule 2 will be evaluated as follows. If the bank performed the transaction successfully, the ticket order in the local control state is updated accordingly and a buying request is forwarded to the seller.

Otherwise, the buyer is informed of the failure, and the order information is removed from the control state. It is worth mentioning that in certain electronic marketplaces the anonymity of the buyer is of great interest: a buyer may not want the seller to know its identity due to the sensitivity of the type of products to be purchased. This type of identity privacy protection could be easily achieved using the same approach.

4.4 Money back guarantee

An electronic marketplace should be able to provide the same money back guarantees as the traditional physical marketplace. Due to various reasons (change of plans or dissatisfaction with the merchandise) a buyer in the ALTM marketplace should be able to cancel a transaction by returning the ticket within a certain time period following the purchase. The buyer should be guaranteed to receive its original payment minus a service fee.

Figure 8 presents the exchange of messages for achieving the money back guarantee, and Fig. 9 presents the fragment of the ALTM law that handles it. According to Rule 1, when a *ticket_transfer* message arrives at the buyer, an *obligation* is imposed at its controller T_b. This obligation has the following meaning: "Remove the rights of the buyer to return the ticket after the grace period t has elapsed." In Rule 2, when the buyer sends a *return_ticket* message back to the seller, T_b will check whether the obligation on this ticket is still pending. If this is the case, i.e. the ticket is within the grace period, the message is forwarded to the seller, and the pending obligation is repealed. Otherwise, if the grace period has passed, the buyer is not allowed to cancel the transaction anymore: T_b will drop the ticket return message and it will inform the

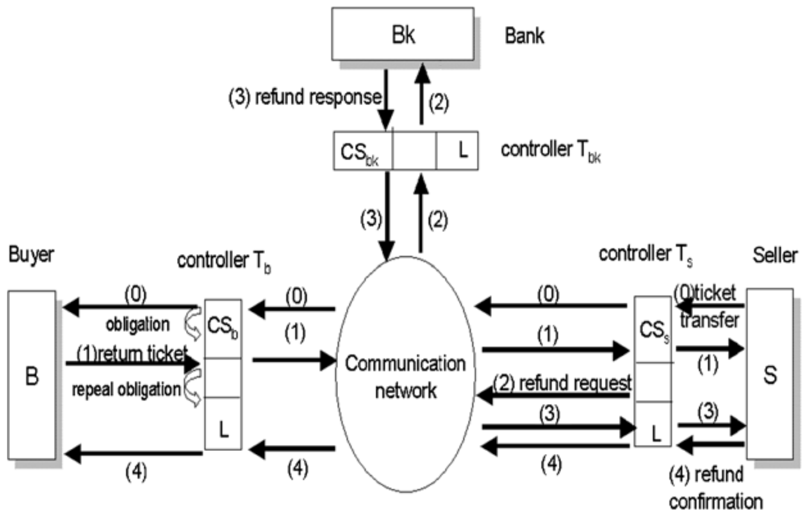

Fig. 8 Money-back guarantee

```
/* When a ticket purchase confirmation arrives at the buyer, an obligation is imposed at
$T_b$ of the buyer. The ticket can be returned anytime before the obligation is up. */

R1: upon arrived(S, ticket_transfer(ticket), B)
        do ImposeObligation(ticket, grace_period)
        do Deliver


/* If the buyer wants to return the ticket, $T_b$ determines whether the ticket is within the
grace period by checking the pending obligation. */

R2: upon sent(B, return_ticket(ticket), S)
        if (CS hasObligation(ticket))
            do RepealObligation(ticket)
            do Forward
        else
            do Deliver ("time_to_return_exceeded")

/* When a return_ticket message arrives at the controller of the seller $T_S$, a refund request is forwarded to the bank automatically.
The message is also delivered to the seller. */

R3: upon arrived(B, return_ticket(ticket), S)
        do Forward(B, ticket, Bank)
        do Deliver
```

Fig. 9 Money-back guarantee: fragment of *ALTM law*

buyer. According to Rule 3, when the return is allowed and the message arrives at the controller of the seller, T_s will send the refund request to the bank after subtracting a service charge from the original payment.

4.5 Reputation tracking

Buyers often use a variety of information when deciding what are the trading partners they are willing to deal with. An example is the reputation of a seller, as a measure of its past business conduct. Reputation is widely used in both traditional and electronic

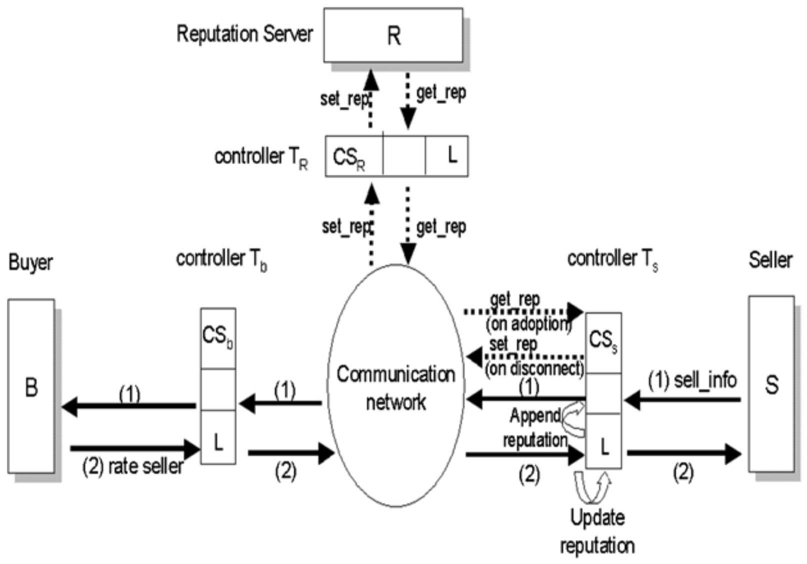

Fig. 10 Reputation tracking

marketplaces. Most often, the reputation tracking mechanism is highly centralized, as is the case of the Better Business Bureau (BBB), or its electronic counterparts such as CNET [20], adding a considerable overhead to each transaction. In ALTM marketplace, the reputation of the sellers has to be tracked and reported in a mostly decentralized manner. In ALTM marketplace, the reputation of each seller is stored in its own controller T_S as presented in Fig. 10. The seller itself cannot change the reputation. The reputation is updated on feedback from buyers. Moreover, every message from a seller to a buyer includes the seller's reputation. Figure 11 presents the fragment of ALTM law that controls the update and use of the reputation.

According to Rule 3, when a seller sends a *sell_information* message to a buyer, the controller T_S of the seller automatically appends the seller's reputation to the message. After a buyer completes a transaction, it is allowed to rate the seller by sending a *rate_seller* message to it. According to Rule 4, when the *rate_seller* message arrives at the seller's controller, T_S updates the reputation accordingly.

The *ALTM Law* defines the *reputation* as a pair (*seniority*, *rating*), where *seniority* indicates the number of past transactions a seller has carried out in this community, and *rating* is the accumulated service rate of a seller. The *seniority* grows by 1 at the completion of each service, and *the rating* is updated after each completed transaction. At the completion of each transaction, a buyer can rate the seller with a rating value in a range of 0 to 10, 10 representing an excellent service.

A challenge posed by this approach is to maintain the long-term reputation for sellers that operate intermittently, and to prevent a seller from setting up multiple clones in order to disguise a negative reputation. In order to address this challenge, we have devised a Reputation Server (RS) entity. RS is responsible for maintaining the reputation of sellers while they are off-line, and for ensuring that *a single* agent is operating in the ALTM marketplace for every authenticated seller. Initially, when

```
/* The Reputation Server RS returns a get_reputation message to a newly adopted seller. When this message arrives at the seller,
its reputation is updated and a role(seller) term is set up in its control state. If RS already has an active record of that seller, it
returns an invalidation answer, and the agent is destroyed. */

R1: upon arrived(RS, get_reputation(ans), S)
            if (CS has role(_))
                  return
            if (ans.value = invalid )
                  do Deliver(seller_already_active)
                  do Quit
            else
                  do Add(role(seller))
                  do UpdateReputation(ans.value)
                  do Deliver


/*When quitting the community, the seller saves its reputation in the Reputation Server*/

R2: upon disconnected()
                  do Forward(S,set_reputation(reputation),RS)
                  do Quit


/* When a seller responds to a buyer with information about a ticket, T_s appends the seller's reputation to the message. */

R3: upon sent(S, sell_info(ticket), B)
            if (CS has role(seller))
                  do Forward(S, sell_info(ticket, reputation), B)


/* Upon a ticket purchase, the buyer can rate the seller's service. T_s will update the seller's reputation in CS. */

R4: upon arrived(B, rate_seller(rating), S)
                  do UpdateReputation(rating)
```

Fig. 11 Reputation tracking: fragment of *ALTM law*

a seller first joins the ALTM marketplace and an LGI agent is created on its behalf, the ALTM law mandates the seller to registers with RS and to retrieve its previously stored reputation. Rule 1 of Fig. 3 has previously shown how the controller of the seller initiates a *get_reputation* message during the seller's adoption event.

Based on its records, the Reputation Server sets up an initial reputation, or it retrieves an already saved reputation, and forwards it to the seller. If the Reputation Server has knowledge of another LGI agent operating on behalf of the same seller, it returns a negative answer. According to Rule 1 in Fig. 11, when the seller receives the *get_reputation* answer from RS, a new pair (*seniority*, *rating*) is set in the control state, and the seller receives a *role*(*seller*) term, enabling it to subsequently operate in the market. When the answer indicates that there exists another operating agent on behalf of the same seller, the agent attempting the setup is destroyed immediately. Note that the Reputation Server uniquely identifies a seller using the previous *Authentication of Identity* mechanism.

Further, when a seller quits the ALTM marketplace effectively destroying its agent, T_s will send its reputation to the Reputation Server, by issuing a *set_reputation* message, as shown in Rule 2 of Fig. 11. When RS receives this message, it will also update its records to reflect the fact that the marketplace has no more active agents on behalf of this seller.

Note that, although the Reputation Server in our scheme is centralized, it is not involved in the routine transactions, but only when a seller adopts or quits its controller;

the reputation is maintained in a distributed manner during the normal operations. Additionally, this approach gives full credit to a seller that has conducted business for a long time and has maintained a good reputation in the marketplace.

Note also that our formula of computing the reputation is not meant to be comprehensive. A number of other factors can be taken into account when computing one's reputation. In particular, the credibility factor of a source, and a history of past interaction can address well-known issues, such as the coalitions among traders, as shown in [10, 11]. Such mechanisms can easily be adopted in the ALTM law, among many other provisions that a law of a DEM may be written to establish.

4.6 Conformance to the governmental laws

In the previous sections we have identified a number of rules whose enforcement enables a secure and trusted framework for trading airline tickets. Additional rules, however, might have a great impact on the functionality of a particular ALTM marketplace, thus their enforcement might become necessary in certain circumstances. Examples of such rules are the governmental laws.

One expects that every marketplace operates under a certain jurisdiction, either international law, or the law of a certain country. Below are several examples of governmental rules that a marketplace might have to enforce:

- The buyers of certain products are to be of a certain minimum age.
- The sellers are to report any sale to a government agency, such as the revenue service, in order to facilitate the collection of the Sales Tax or VAT.
- The sellers have to report the names of the buyers of airline tickets to a governmental agency, such as the Transportation Security Administration (an example of such recent legislation is the Multi-state Anti-Terrorism Information Exchange (MATRIX) program.)

Even though the implementation of such rules might appear similar to the rules previously discussed in this paper, they pose problems of a different nature, thus they require further discussion. First, for the writer of the marketplace law, the addition of such rules may increase the law to an un-manageable size, making it unintelligible, thus prone to errors. Additionally, since the rules to be incorporated in the law of the marketplace are derived from multiple and heterogeneous sources, it is hard to combine them in a single and *consistent* policy. This becomes more pregnant when the rules are not entirely independent, posing the danger of interference. For example, if the ALTM marketplace would employ a rule that insures the anonymity of the buyers, this rule might conflict with the previously defined governmental rule which requires that the name of the buyers, or their age be authenticated. Second, from the point of view of the government, it is hard to both identify the rules responsible for the governmental laws, and to verify that those rules actually enforce the desired behavior. A modularization scheme and a precedence mechanism have to be employed in this case.

All these considerations lead us to organize the rules in a DEM into a modular hierarchy of laws. The LGI law hierarchy, introduced in Sect. 3.2.4, ensures that the entire policy of the marketplace is consistent, and that certain rules (like governmental rules) have a higher priority and can take precedence over other rules.

```
/* When the seller issues a ticket_transfer message, the top law delegates the event to the
subordinate law. If the subordinate law decides that the transaction is invalid, it does nothing. In the
case of a successful transaction, the top law reports the sales tax to a  Governmental Agency*/

R1: upon sent(S, ticket_transfer(ticket), B)
                                   do Delegate
                    if not ( transaction = successful)
                                   return
                    else
                                   do Forward(S, sales_tax(ticket.price), GA)
```

Fig. 12 Reporting of the sales tax to a governmental agency: fragment of *ALTM law*

Concretely, Fig. 12 implements the example rule that mandates reporting of ticket sales to a governmental agency. This rule has to be separated from, and should take precedence over, the ordinary rules of ALTM. Thus, the rule is to be part of a law called *Governmental Law* (*G Law*); the *ALTM Law* is to inherit from, and conform to, the *G Law*. Whenever a seller submits *a ticket_transfer* message, the controller first evaluates the G Law, as shown in Fig. 12, Rule 1. This rule proceeds as follows. First it delegates the event to the subordinate, ALTM law (do Delegate). In this case, Rule 3 in Fig. 5 (i.e. the treatment of the ticket authenticity) is evaluated in order to decide whether the sale of such tickets is valid under the current marketplace rules. If Rule 3 allows the transaction to proceed, the G Law will detect the success of the transaction and it will report the sales tax to a Government Agency. Otherwise, nothing will happen. The interaction between the top, Government Law and the subordinate rules is such that G Law will always take precedence over the other rules in the ALTM law, and it will always be enforced. The ordinary rules in the ALTM law, in turn will only be enforced as long as they do not contradict the Government Law.

5 Conclusion

The concepts of *Decentralized Electronic Marketplace* (DEM) proposed in this paper is bound to no physical location that might correspond to the marketplace, nor to a virtual location, that might take the form of a central manager, or mediator for all transactions. Yet, it deserves the name “marketplace” in that it provides a single and unifying law that governs all the transactions made through it—in some analogy to the laws and customs that govern traditional marketplaces.

The law of a given DEM is explicitly defined, and visible to all its participants. Moreover, the law is strictly enforced, via the LGI mechanism, in a completely decentralized manner. This makes such a marketplace easy to launch, essentially by writing its law; easy for buyers or sellers to engage in, simply by locating an authenticated controller (using a provided controller-service), and adopting it with the law of the DEM; and easy to scale, due to its decentralized nature, free from any central mediator. Note that every participant in a DEM can obtain the text of the law that regulates the marketplace. Even if the participant himself does not understand this law, he can present it to a judicial court in the case of a dispute. This is in contrast with the case of traditional marketplaces, such as EBay, whose rules are implicit in the code of the server, and they are not available to the scrutiny of the participants.

We provide supporting evidence to the efficacy of this concept by presenting a study of a DEM devoted to the Airline Ticket Trading (ALTM) marketplace, governed by the *ALTM Law*. This law has been described here using an intuitive pseudo-code, it has been fully defined, implemented, and experimented with, and it is made available on our web site using two alternative programming languages.

It should, however, be pointed out that although the LGI mechanism, on which our concept of DEM is based, has been fully implemented, and although it has been working in our lab for several years now, for a DEM to be usable by buyers and sellers dispersed throughout the Internet, the LGI middleware needs to be commercially, and widely, deployed. Such deployment is beyond our capacity, and we can do here no more then advocate its wide-scale deployment.

Acknowledgement We thank the anonymous reviewers for the insightful comments that helped us improve the quality of the paper.

References

1. Open Market (1998). *Internet commerce: the open market transact solution*. Technical White Paper, Open Market, Inc., July 1998. Available from www.openmarket.com.
2. Wurman, P., Wellman, M., & Walsh, W. (1998). The Michigan Internet AuctionBot: a configurable auction server for human and software agents. In K. P. Sycara & M. Wooldridge (Eds.), *Proceedings of the 2nd international conference on autonomous agents (agents'98)* (pp. 301–308). New York: ACM Press.
3. Lacoste, G. (1997). SEMPER: A security framework for the global electronic marketplace. *Comtec – the Magazine for Telecommunications Technology*.
4. Minsky, N. (1991). The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, February 1991.
5. Minsky, N. (2005). Law-governed interaction (LGI): a distributed coordination and control mechanism (An introduction and a reference manual). Technical Report, Rutgers University, Available at: http://www.moses.rutgers.edu/documentation/manual.pdf. June 2005.
6. Serban, C. The LGI Web-Site. Available at: http://www.moses.rutgers.edu.
7. Chen, Y., Serban, C., Zhang, W., & Minsky, N. (2005). Towards a decentralized and secure electronic marketplace. In *Proc. of the 5th IADIS e-commerce conference (IADIS e-commerce 2005)*. Porto, Portugal, December 2005.
8. Schmees, M. (2003). Distributed digital commerce. In *Proceedings of the 5th international conference on electronic commerce (ICEC2003)* (pp. 131–137). ACM: Pittsburgh, PA, USA.
9. Wainder, M. (1996). Development of a secure electronic marketplace for Europe. In *Fourth European symposium on research in computer security (ESORICS)*. Rome, Italy, September 1996.
10. Xiong, L., & Liu, L. (2003). A reputation-based trust model for peer-to-peer eCommerce communities. In *Proc. of the fourth ACM conference on electronic commerce (EC'03)*. San Diego, June, 2003.
11. Xiong, L., & Liu, L. (2004). Reputation and trust in mobile commerce. In *Advances in security and payment methods for mobile commerce*. Idea Group, November, 2004. ISBN 1591403456.
12. Fontoura, M., Ionescu, M., & Minsky, N. (2005). Decentralized peer-to-peer auctions. *Journal of Electronic Commerce Research*, January 2005.
13. Ao, X., & Minsky, N. (2003). Flexible regulations of distributed coalitions. In *Proceedings of the eighth European symposium on research in computer security (ESORICS)*. Norway, October 2003.
14. Minsky, N., & Ungureanu, V. (2000). Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, *9*(3), 273–305.
15. Ungureanu, V., & Minsky, N. (2000). Establishing business rules for inter-enterprise electronic commerce. In *Proceedings of the fourteenth international symposium on distributed computing (DISC 2000)* (pp. 179–193). LNCS 1914, Toledo, Spain, October 2000.
16. Schneier, B. (1996). *Applied cryptography*. New York: Wiley.

17. Ellison, C. (1999). The nature of a usable PKI. *Computer Networks*, *31*, 823–830.
18. Ao, X., & Minsky, N. (2003). Flexible regulation of distributed coalitions. In *Proc. of the 8th European symposium on research in computer security (ESORICS)*, Gjøvik, Norway, October 2003.
19. Minsky, N., & Murata, T. (2004). On manageability and robustness of open multi-agent systems. In C. Lucena, A. Garcia, A. Romanovsky, J. Castro, & P. Alencar (Eds.), *LNCS: Vol. 2940. Computer security, dependability, and assurance* (pp. 189–206). Berlin: Springer.
20. Smith, R. G. (1988). The contract net protocol: high-level communication and control in a distributed problem solver. In *Distributed artificial intelligence* (pp. 357–366). San Francisco: Morgan Kaufmann.

Constantin Serban has received a B.S. degree in Computer Science and Engineering from the Polytechnic University of Bucharest, Romania, in 1996, a M.S. in Computer Science and Engineering from the same university in 1997, and a Ph.D. in Computer Science from Rutgers University in 2008. His research interests are in security, dependability, and software engineering for large and distributed systems. His primary research area is the policy based enforcement of access control in distributed systems. He also worked on software engineering, where he addressed the enforcement of software architectures on large systems, both monolithic and distributed. Currently Constantin Serban is a senior researcher with the Policy-Based Network Management Group in the Applied Research Department at Telcordia Technologies. He is a member of the IEEE and ACM.

Yingying Chen received the B.S. degree in Physics from Nanjing University, China, in 1991, the M.S. degree in Computer Science from North Carolina State University in 1994, and the Ph.D. in Computer Science from Rutgers University in 2007. She is currently an assistant professor in the Department of Electrical and Computer Engineering at Stevens Institute of Technology. Her research interests include wireless and system security, wireless networking, and distributed systems. Prior to joining Stevens Institute of Technology, she was with Bell Laboratories and the Optical Networking Group, Lucent Technologies. She also worked as a system researcher for embedded networks at Alcatel Network Systems. Dr. Chen received the IEEE Outstanding Contribution Award from IEEE New Jersey Coast Section each year from 2000 to 2005. She is the recipient of the Best Technological Innovation Award from the 3rd International TinyOS Technology Exchange in 2006. She is a member of the IEEE and ACM. She has been an executive officer for IEEE Computer and Communication Joint Chapter at New Jersey Coast Section since 2000, and has served on the technical programs for several IEEE/ACM conferences on wireless networking and security.

Wenxuan Zhang is a Ph.D. candidate in computer science at Rutgers University. His research interests include distributed systems, software engineering and electronic commerce. His current work centers on improving large-scale software system dependability with Law-Governed Interaction (LGI), a coordination and control mechanism of distributed systems. Wenxuan received his B.S. degree in computer science from University of Science and Technology of China (USTC) in 2000, and his M.S. degree in computer science from Rutgers University in 2002. Contact him at wzhang@cs.rutgers.edu.

Naftaly Minsky obtained his Ph.D. in Theoretical Physics, from the Hebrew University of Jerusalem. He taught Physics and Computer Science in Hebrew University, and Computer Science in the University of Minnesota and in Rutgers University. His current research interests include making large scale distributed systems simpler, more manageable, and more secure; and he has been studying application domains such as e-commerce, enterprise systems, and virtual enterprises.